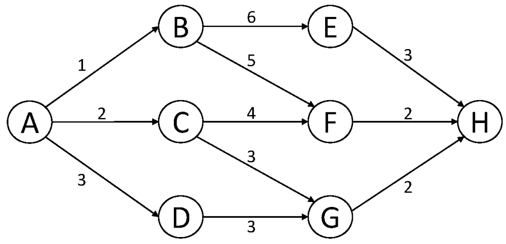
Dynamic programming

- Closed loop formulation of optimal control
- Intuitive methods of solution
- Guarantees global optimality
- Iteratively solves the problem starting at the end-time

'Life can only be understood backwards;
but it must be lived forwards'

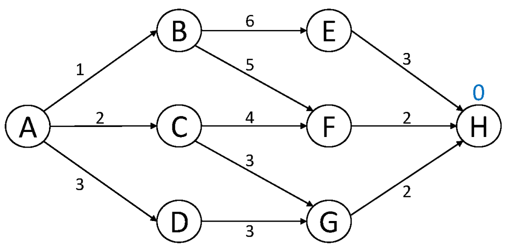
Kierkegaard

Example: Shortest path



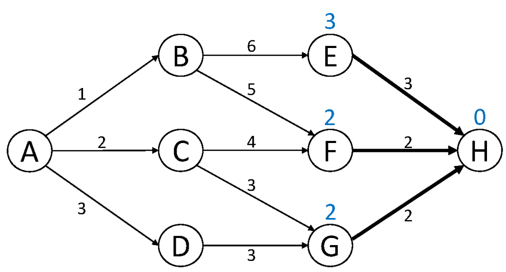
As an example we try to find the shortest path from “A” to “H” in the above graph.

Example: Shortest path



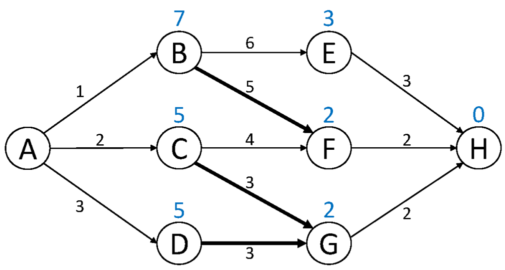
We proceed with backward induction. Once the final node is reached the remaining cost is 0.

Example: Shortest path



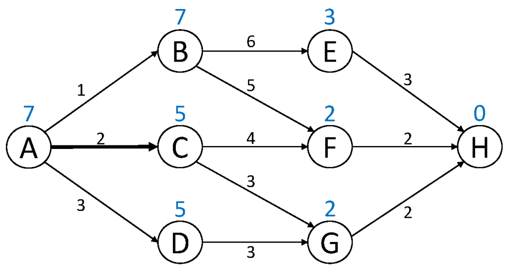
Knowing the cost at “H” to be 0, costs of getting from “E”, “F” and “G” to “H” are easily computed.

Example: Shortest path



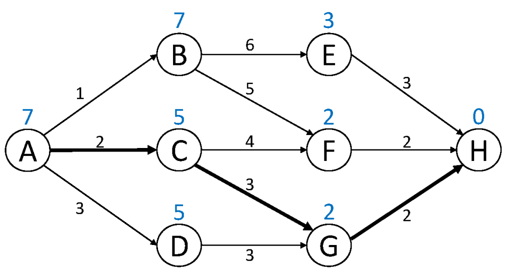
Now the optimal “cost-to-go” at “E”, “F” and “G” can be used to get the optimal “cost-to-go” at “B”, “C” and “D”.

Example: Shortest path



In the next step we arrive at the origin.

Example: Shortest path



The procedure also gives us the optimal path.

Basic problem formulation: The system

- First we assume that the system is in discrete time

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1$$

where x_k is the state $u_k \in U(x_k)$ is the control.

- Feedback-control implies $u_k = \mu_k(x_k)$
- In closed-loop form the system can thus be written

$$x_{k+1} = f_k(x_k, \mu_k(x_k))$$

Basic problem formulation: The cost

- We let $\mu = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ and assume that we have an additive cost

$$J_\mu(x_0) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k))$$

- Total cost $J_\mu(x_0)$ is a function of both initial state x_0 and feedback law μ
- N is the horizon of the problem
  - Finite-horizon: $N < \infty$
  - Infinite-horizon: $N = \infty$

Basic formulation: Minimal cost and optimal strategy

- An optimal policy μ^* is one that minimizes $J_\mu(x_0)$ (for all x_0)

$$J_{\mu^*}(x_0) = \min_{\mu \in \Pi} J_\mu(x_0)$$

optimization is performed over the set Π, of admissible control policies

- For deterministic problems a control is admissible whenever

$$u_k = \mu_k(x_k) \in U(x_k)$$

The principle of optimality

Let $\mu^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be an optimal policy for the basic problem and assume that when applying μ^*, a given state x_i occurs at time i, when starting at x_0.

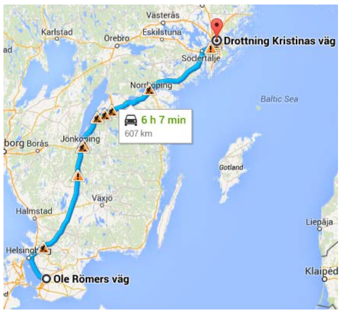
Consider the subproblem whereby we are in state x_i at time i and wish to minimize the "cost-to-go" from time i to time N

$$g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k)).$$

Principle of optimality

The truncated policy $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ is optimal for the subproblem starting from x_i at time i.

Principle of optimality



- Google maps fastest route from LTH to KTH passes through Jönköping
- Subpath starting in Jönköping is the fastest route from Jönköping to KTH

The dynamic programming algorithm

Let

$$V_k(x_k) = g_N(x_N) + \sum_{j=k}^{N-1} g_j(x_j, \mu_j^*(x_j))$$

so that $V_k(x_k)$ is the optimal "cost-to-go" from time k to time N

The Bellman equation

For every initial state x_0, the optimal cost $J^*(x_0)$ is given by the last step in the following backward-recursion.

$$V_k(x_k) = \min_{u_k \in U_k(x_k)} [g_k(x_k, u_k, w_k) + V_{k+1}(f_k(x_k, u_k))]$$

$$V_N(x_N) = g_N(x_N)$$

We get the optimal control "for-free"

$$\mu_k^*(x_k) = \arg\min_{u_k \in U_k(x_k)} [g_k(x_k, u_k, w_k) + V_{k+1}(f_k(x_k, u_k))]$$

Managing spending and saving

Example

An investor holds a capital sum in a building society, which gives an interest rate of $\theta \times 100\%$ on the sum held at each time $k = 0, 1, \ldots, N-1$. The investor can chose to reinvest a portion u of the interest paid which then itself attracts interest. No amounts invested can ever be withdrawn. How should the investor act so as to maximize total reward by time $N-1$?

- We take as the state x_k the present income at time $k = 0, 1, \ldots, N-1$ and let $u_k \in [0, 1]$ be the fraction of reinvested interest, hence

$$x_{k+1} = x_k + \theta u_k x_k =: f(x_k, u_k)$$

- The reward is $g_k(x, u) = (1-u)x$ and $g_N(x, u) \equiv 0$.

Managing spending and saving

- The optimality equation is $V(N, x) = 0$,

$$V(k, x) = \max_{0 \le u \le 1} \{(1-u)x + V(k+1, (1+\theta u)x)\}, \quad k = 0, 1, \ldots, N-1$$

- We get

$$V(N-1, x) = \max_{0 \le u \le 1} \{(1-u)x + 0\} = x$$

$$V(N-2, x) = \max_{0 \le u \le 1} \{(1-u)x + (1+\theta u)x\}$$

$$= \max_{0 \le u \le 1} \{2x + (\theta - 1)ux\} = \max\{2, 1+\theta\}x = \rho_2 x$$

- Guess: $V(N-s+1, x) = \rho_{s-1}x$, then

$$V(N-s, x) = \max_{0 \le u \le 1} \{(1-u)x + \rho_{s-1}(1+u\theta)x)\}$$

$$= \max\{1 + \rho_{s-1}, (1+\theta)\rho_{s-1}\}x = \rho_s x$$

Managing spending and saving

- We have thus verified that $V(N-s, x) = \rho_s x$, and found the recursion

$$\rho_s = \rho_{s-1} + \max\{1, \theta\rho_{s-1}\}$$

- Together with $\rho_1 = 1$ this gives

$$\rho_s = \begin{cases} s & \text{for } s \le s^* \\ s^*(1+\theta)^{s-s^*} & \text{otherwise.} \end{cases} \qquad s^* = \lceil 1/\theta \rceil$$

- The optimal policy is then

$$u_k = \begin{cases} 1 & \text{for } k < N - s^* \\ 0 & \text{for } k \ge N - s^*. \end{cases}$$

Continuous time optimal control: The HJB-equation

- So far we have only considered the discrete time case
- Dynamic programming can also be applied in continuous time, this leads to the Hamilton-Jacobi-Bellman (HJB) equation:
- Benefits over PMP:
  - + Gives closed-loop optimal control in continuous time
  - + Sufficient condition of optimality
- Drawbacks:
  - – Requires solving a highly non-linear PDE
  - – Well-posedness of the PDE problem proved only in the '80s

Continuous time problem formulation

- In continuous time the system is given by

$$\dot{x}(t) = f(x(t), u(t)), \quad t \in [0, T]$$

with $x(0) = x_0$ and $u(t) \in U(x(t))$, for all $t \in [0, T]$.

- We define the cost as

$$J(x_0) = \phi(x(T)) + \int_0^T L(x(t), u(t))dt$$

- With optimal "cost-to-go" from (t, x)

$$V(t, x) = \min_u \left\{ \phi(x(T)) + \int_t^T L(x(t), u(t))dt \right\}$$

The HJB-equation: Informal derivation

- divide $[0, T]$ into N subintervals of length $\delta = T/N$
- Let $x_k = x(k\delta)$ and $u_k = u(k\delta)$, for $k = 0, 1, \dots, N$ and approximate the system by

$$x_{k+1} = x_k + f(x_k, u_k)\delta, \quad k = 0, 1, \dots, N.$$

- The optimal "cost-to-go" is approximated by

$$V(k\delta, x) = \min_{u_0, \dots, u_{N-1}} [\phi(x_N) + \sum_{k=0}^{N-1} L(x_k, u_k)\delta]$$

The HJB-equation: Informal derivation

- Dynamic programming now yields

$$V(k\delta, x) = \min_{u \in U} [L(x, u)\delta + V((k+1)\delta, x + f(x, u)\delta)],$$
$$V(N\delta, x) = \phi(x).$$

- For small δ we get (with $t = k\delta$)

$$V(t + \delta, x + f(x, u)\delta) \approx V(t, x) + V_t(t, x)\delta + \nabla_x V(t, x) \cdot f(x, u)\delta$$

- Inserting this in the DP equation gives

$$V(t, x) \approx \min_{u \in U} [L(x, u)\delta + V(t, x) + V_t(t, x)\delta + \nabla_x V(t, x) \cdot f(x, u)\delta]$$

The HJB-equation

The Hamilton-Jacobi-Bellman equation

For every initial state x_0, the optimal cost is given by $J^*(x_0) = V(0, x_0)$ where $V(t, x)$ is the solution to the PDE

$$V_t(t, x) = -\min_{u \in U} [L(x, u) + \nabla_x V(t, x) \cdot f(x, u)]$$
$$V(T, x) = \phi(x)$$

As before the optimal control is given in feedback form by

$$\mu^*(t, x) = \arg\min_{u \in U} [L(x, u) + \nabla_x V(t, x) \cdot f(x, u)]$$

Example: The HJB-equation

Example

Consider the simple example involving the scalar system

$$\dot{x}(t) = u(t),$$

with the constraint $|u(t)| \leq 1$ for all $t \in [0, T]$ and the cost

$$J(x_0) = \frac{1}{2}(x(T))^2.$$

- The HJB equation for this problem is

$$V_t(t, x) = -\min_{|u(t)| \leq 1} [V_x(t, x)u]$$

with terminal condition $V(T, x) = x^2/2$.

Example: The HJB-equation

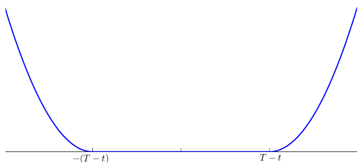
- An optimal control for this problem is

$$\mu(t, x) = \begin{cases} 1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x > 0 \end{cases}$$

- The optimal "cost-to-go" with this control is

$$V(t, x) = \frac{1}{2}(\max\{0, |x| - (T - t)\})^2$$

Example: The HJB-equation



- For $|x| > T - t$ we have $V(t, x) = 1/2(|x| - (T - t))^2$, hence

$$V_t = |x| - (T - t)$$
$$\min_{|u(t)| \leq 1} [V_x(t, x)u] = -\text{sgn}(x)V_x(t, x) = -\text{sgn}(x)^2(|x| - (T - t))$$
$$= -(|x| - (T - t))$$

- For $|x| \leq T - t$ we have $V(t, x) = 0$ and the HJB equation holds trivially

Infinite horizon problem

Assume that the final cost is $\phi(x(T)) = 0$ and the final time $T \to +\infty$, and that there exists some control such that the total cost remains bounded in the limit. Hence, we want to solve

$$\min_u \int_0^{+\infty} L(x(t), u(t))dt\,, \qquad x(0) = x_0$$

It is intuitive that the cost-to-go from (x, t)

$$V(x,t) = \min_u \int_t^T L(x(t), u(t))dt = V(x)$$

does not depend on the initial time but only on the initial state.

The HJB equation then becomes

$$0 = \min_u \left[L(x,u) + \nabla_x V(x) \cdot f(x,u)\right]$$

(Observe that, for scalar problems, this is an ODE!)

Infinite horizon problem: example

$$\min_u \int_0^{+\infty} (x^4(t) + u^4(t))dt\,, \qquad x(0) = x_0$$

From the stationary HJB eqn we get

$$0 = \min_u \left\{x^4 + u^4 + V_x(x) \cdot u\right\}$$

and putting the derivative with respect to u equal to 0

$$x^4 = 3\left(\frac{1}{4}V_x(x)\right)^{4/3}$$

which gives $V_x(x) = \pm 4(\frac{1}{3})^{3/4}x^3$ and the $+$ sign should be chosen to have V positive definite)since L is. This gives the optimal feedback control law

$$u^*(x) = -(\frac{1}{4}V_x(x))^{1/3} = -(\frac{1}{3})^{1/4}x$$

Dynamics Programming for LQ control

Consider the optimal feedback control problem for an LTI system $\dot{x} = Ax + Bu$ with cost

$$J = \int_0^T \left(x'(t)Qx(t) + u'(t)Ru(t)\right)dt + x(T)'Mx(T)$$

where Q, R, M are symmetric positive definite. The HJB eqn reads

$$0 = \min_u \left\{x'Qx + u'Ru + V_t + V_x'(Ax + Bu)\right\}$$

with final time condition $V(T, x) = x'Mx$.

Dynamics Programming for LQ control

With the ansatz $V(x,t) = x'P(t)x$ with symmetric $P(t)$, we get that the optimal control is in the form

$$u^* = -R^{-1}B'Px$$

and $P = P(t)$ satisfies the following differential eqn

$$\dot{P} = -PA - A'P - Q + PBR^{-1}B'P \qquad P(T) = M$$

which is called the differential Riccati equation (DRE).

For the infinite horizon problem this reduces to

$$0 = -PA - A'P - Q + PBR^{-1}B'P$$

which is called the algebraic Riccati equation (ARE).

Bonus: Dynamic programming and randomness

- So far we have only considered deterministic systems
- For deterministic systems open-loop and closed-loop control coincide
  - Minimizing over admissible policies $\mu = \{\mu_0 \dots, \mu_{N-1}\}$ equivalent to minimizing over control vectors $\{u_0, \dots, u_{N-1}\}$
  - Given μ, future states are perfectly predictable through

    $$x_{k+1} = f_k(x_k, \mu_k(x_k)), \quad k = 0, 1, \dots, N-1$$

  - Corresponding controls perfectly predictable through

    $$u_k = \mu_k(x_k)$$

- When introducing randomness in the state evolution, closing the loop becomes important

Problem with randomness: The system

- We assume that the system is in discrete time but add randomness

  $$x_{k+1} = f_k(x_k, u_k, w_k)$$

  where x_k is the state $u_k \in U(x_k)$ is the control and w_k is a noise term.
- The distribution of the noise term w_k only depends on the past through x_k and u_k
- In closed-loop form the system can thus be written

  $$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k)$$

Basic problem formulation: The cost

- In the random case we get the cost

  $$J_\mu(x_0) = E\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right]$$

  where expectation is taken over the random variables x_k and w_k
- Expected cost $J_\mu(x_0)$ is a function of both initial state x_0 and feedback law μ

Minimal cost and optimal strategy

- An optimal policy μ^* is a policy that minimizes $J_\mu(x_0)$ (for every x_0)

  $$J_{\mu^*}(x_0) = \min_{\mu \in \Pi} J_\mu(x_0)$$

- Optimization is performed over the set, Π, of admissible controls
  - $u_k \in U(x_k)$, for all x_k
  - u_k does not depend on future events
- Optimal control is in feedback-form $u_k^* = \mu_k^*(x_k^*)$

The value of information

Two chess players play a two round chess match. Winning one round gives 1 point, drawing gives 1/2 and losing gives 0. If the score after the two rounds is tied the match will be decided by sudden death.
Player 1 has the opportunity of adapting his strategy by selecting to play either *timid* or *bold*,

- Timid: Draws with probability p_d and loses with probability $1 - p_d$ (no chance of winning)
- Bold: Wins with probability p_w and loses with probability $1 - p_w$ (no chance of drawing)

Two round chess match

Player 1 is thus faced with the problem of finding the strategy that maximizes his probability of winning the match.

Open-loop strategy

With an open-loop strategy Player 1 has to decide beforehand how to play in each round.

1. Timid-timid: Probability $p_d^2 p_w$ of winning the match
2. Bold-bold: Probability $p_w^2 + 2p_w^2(1-p_w) = p_w^2(3-2p_w)$ of winning the match
3. Timid-bold: Probability $p_d p_w + (1-p_d)p_w^2$ of winning match
4. Bold-timid: Probability $p_w p_d + p_w^2(1-p_d)$ of winning match

$$\text{Open-loop win probability} = \max(p_w^2(3-2p_w), p_w p_d + p_w^2(1-p_d)) = p_w^2 + p_w(1-p_w)\max(2p_w, p_d)$$

Optimal open loop strategy:

- $p_d > 2p_w$: Timid-bold or bold-timid
- $p_d < 2p_w$: Bold-bold
- $p_d = 2p_w$: All except timid-timid are optimal

Closed-loop strategy

Here we start with a bold strategy in the first round and choose

1. Bold-timid: If score is 1-0 after Round 1
2. Bold-bold: If score is 0-1 after Round 1

$$\text{Closed-loop probability of win} = p_w p_d + p_w^2(1-p_d) + (1-p_w)p_w^2 = p_w^2 + p_w(1-p_w)(p_w + p_d)$$

Comparing with the open-loop case gives

$$\begin{aligned}\text{Value of information} =& p_w^2 + p_w(1-p_w)(p_w+p_d) \\ & - p_w^2 - p_w(1-p_w)\max(2p_w, p_d) \\ =& p_w(1-p_w)\min(p_w, p_d - p_w)\end{aligned}$$

The dynamic programming algorithm

Now,

$$V_k(x_k) = E\left[g_N(x_N) + \sum_{j=k}^{N-1} g_j(x_j, \mu_j^*(x_j), w_j)\right]$$

The Bellman equation

For every initial state x_0, the optimal cost $J^*(x_0)$ is given by the last step in the following backward-recursion.

$$V_k(x_k) = \min_{u_k \in U_k(x_k)} E\left[g_k(x_k, u_k, w_k) + V_{k+1}(f_k(x_k, u_k, w_k))\right]$$
$$V_N(x_N) = g_N(x_N)$$

We get the optimal control “for-free”

$$\mu_k^*(x_k) = \arg\min_{u_k \in U_k(x_k)} E\left[g_k(x_k, u_k, w_k) + V_{k+1}(f_k(x_k, u_k, w_k))\right]$$

Example: Selling an asset

Optimal asset selling

Consider a person having an asset that has to sell within N time periods. Every time period he gets a new offer, that he can either accept or reject. These offers are given by a sequence of independent random variables $w_0, w_1, \dots, w_{N-1}$. When the seller accepts an offer he can invest the money at fixed interest rate $r > 0$. The sellers objective is to maximize the revenue at day N.

- We let $u_k = 0$ represent rejecting to k^{th} offer and $u_k = 1$ when accepting offer k
- We also introduce the terminal state T that x_k enters once the asset is sold and get the state equation $x_{k+1} = f(x_k, w_k)$, where

$$f(x_k, w_k) = \begin{cases} T & \text{if } x_k = T \text{ (sold), or if } x_k \neq T \text{ and } u_k = 1 \text{ (sell),} \\ w_k & \text{otherwise.}\end{cases}$$

Example: Selling an asset

- The corresponding reward function may be written as

$$E\left[g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, u_k, w_k)\right]$$

where

$$g_N(x_N) = \begin{cases} x_N & \text{if } x_N \neq T \\ 0 & \text{if } x_N = T.\end{cases}$$

and

$$g_k(x_k, u_k, w_k) = \begin{cases} (1+r)^{N-k} x_k & \text{if } x_k \neq T \text{ and } u_k = 1 \text{ (sell),} \\ 0 & \text{otherwise.}\end{cases}$$

Example: Selling an asset

- This gives the DP algorithm

$$V_N(x_N) = \begin{cases} x_N & \text{if } x_N \neq T \\ 0 & \text{if } x_N = T\end{cases}$$

and

$$V_k(x_k) = \begin{cases} \max\{(1+r)^{N-k} x_k, E[V_{k+1}(w_k)]\} & x_k \neq T \\ 0 & x_k = T.\end{cases}$$

- We thus get the policy

$$u_k = \begin{cases} 1 & \text{if } x_k > \alpha_k \\ 0 & \text{if } x_k < \alpha_k,\end{cases}$$

where

$$\alpha_k = \frac{E[V_{k+1}(w_k)]}{(1+r)^{N-k}}$$

Example: Selling an asset

- Let us now assume that the w_k are identically distributed
- Introduce the functions $G_k(x_k) = (1+r)^{k-N} V_k(x_k)$, hence for $x_N, x_k \neq T$

$$G_N(x_N) = x_N$$
$$G_k(x_k) = \max\{x_k, (1+r)^{-1} E[G_{k+1}(w)]\}$$

and

$$\alpha_k = \frac{E[G_{k+1}(w)]}{1+r}$$

- Now $G_{N-1}(x) \geq G_N(x)$ and if $G_{j+1}(x) \geq G_{j+2}(x)$ then $G_j(x) \geq G_{j+1}(x)$, hence by induction $G_k(x) \geq G_{k+1}(x)$ for $k = 0, \dots, N-1$
- This shows that α_k is a decreasing sequence

Example: Selling an asset

- To compute the sequence α_k we note that $G_k(x_k) = \max\{x_k, \alpha_k\}$, hence

$$\begin{aligned}\alpha_k &= \frac{1}{1+r}E[G_{k+1}(w)] \\ &= \frac{1}{1+r}\alpha_{k+1}P[w \le \alpha_{k+1}] + \frac{1}{1+r}\int_{\alpha_{k+1}}^{\infty} x f_w(x)dx\end{aligned}$$

- Since by definition $\alpha_N = 0$ this gives a recursion for α_k, $k = 1, \dots, N$

Example: Selling an asset

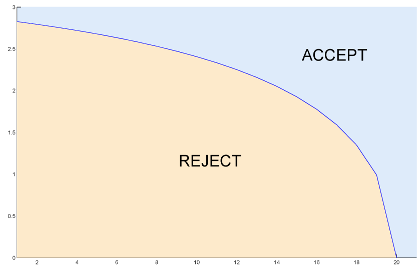
- Assume that w is Exp(1) distributed *i.e.* $f_w(x) = e^{-x}$
- We have $P[w \le \alpha_{k+1}] = 1 - e^{-\alpha_{k+1}}$ and

$$\int_{\alpha_{k+1}}^{\infty} x f_w(x)dx = e^{-\alpha_{k+1}}(\alpha_{k+1} + 1)$$

- This gives the recursion

$$\begin{aligned}\alpha_k &= \frac{1}{1+r}\alpha_{k+1}(1 - e^{-\alpha_{k+1}}) + \frac{1}{1+r}e^{-\alpha_{k+1}}(\alpha_{k+1} + 1) \\ &= \frac{1}{1+r}(\alpha_{k+1} + e^{-\alpha_{k+1}})\end{aligned}$$

Example: Selling an asset



The figure shows the optimal policy for $r = 0.01$ and $N = 20$.

Optimal stopping

- Optimal stopping problems are a special case of the basic problem in which the control can only take two values *e.g.* $\{0, 1\}$ one of which renders the cost (reward) $\phi_k(x)$ and makes the system enter an absorbing terminal state T after which no further cost is incurred
- The Dynamic programming algorithm for optimal stopping problems can be written

$$\begin{aligned}V_N(x_N) &= \phi_N(x_N) \\ V_k(x_k) &= \min\{\phi_k(x_k), E\left[V_{k+1}(f(x_k, w_k))\right]\}\end{aligned}$$

- For optimal stopping problems we can define a set $T_k = \{x : \phi_k(x) < E\left[V_{k+1}(f(x_k, w_k))\right]\}$ called the *termination set*

Optimal stopping: The one-stage look-ahead rule

- Sometimes extracting the optimal policy by backward iteration in the DP algorithm is complex
- For a specific type of problems we do not need to solve the DP however
- Define the set $S = \{(k, x) : \phi_k(x) < E\left[\phi_{k+1}(f(x_k, w_k))\right]\}$
  - If $(k, x_k) \in S$ it is better to stop now than to continue and stop in the next step
- Assume that the set S is *absorbing* in the sense that $(k+1, f(x_k, w_k)) \in S$ whenever $(k, x_k) \in S$
- Then it is optimal to stop iff $(k, x_k) \in S$.